UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/899,868	07/05/2001	Jeanne L. Doyle	79-01-003	3339

7590 02/10/2005

David G. Wille, Esq.
Baker Botts L.L.P.
Suite 600
2001 Ross Avenue
Dallas, TX 75201-2980

EXAMINER	
VU, TUAN A	

ART UNIT	PAPER NUMBER
2124	

DATE MAILED: 02/10/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/899,868	DOYLE ET AL.	
	Examiner	Art Unit	
	Tuan A Vu	2124	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *30 September 2004*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-37* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-37* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to the Applicant's response filed 9/30/2004.

As indicated in Applicant's response, claims 1, 13, 23, 29, 35, and 37 have been amended. Claims 1-37 are pending in the office action.

Claim Rejections - 35 USC § 103

2. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

3. Claims 1-7, 13-18, 23-26, and 37 are rejected under 35 U.S.C. 103(a) as being unpatentable over IBM, "Software Compiler for Analysis and Measuring Programs", 9/1993 (hereinafter Scamp), in view of Leonard, USPN: 5,729,746 (hereinafter Leonard), and further in view of Duan et al., USPN: 6,529,865 (hereinafter Duan).

As per claim 1, Scamp discloses a source code counting system, comprising:

a computer readable storage medium and a software counting tool stored therein and operable to parse a first file containing computer source code to create a token stream in response to one of a plurality of sets of configuration data (e.g. *grammar file* - pg. 123, bottom; *TOKEN COLLECTOR, ...specified in the grammar* - pg. 124-126),

wherein the computer source code file was written in one of a plurality of computer languages that may be processed for the software counting tool (e.g. *independently .. code is written* – pg. 127, 2nd para – Note: a marketable tool to gather metrics necessarily discloses that a variety of programming language can be treated),

create a list of statements in response to the token stream (e.g. *section ... metrics ... local scope ... for each unique token* – pg. 124, 4th para; *token ... flag* - bottom pg. 124, top 8 paras - pg. 125 – Note: flagging code constructs as parsed using a grammar specification is equivalent to setting statements with tag, hence list of statements), and generate a metric value in response to the list of statements(e.g. *source code measurements, how many times* – pg. 127, 2nd para), at least some of the statements comprising a combination of two or more tokens (e.g. the report section having a cumulative type of metrics on tokens reads on comprising a combination of two or more tokens).

But Scamp does not explicitly disclose that the source code counting system is a source code line counting system. But Scamp discloses a token stream obtained from a programming source file as well as a resulting token collector associating tagging/flagging for specific source code constructs (e.g. *Token collector, flags*– pg. 124-125) to generate metrics based on tags or terminal/non-terminal tokens in a specification file (*START, FINISH* -- 1st para , pg. 127); thus, the concept of source code line measurement is strongly implied. Analyzing source code to generate a count for a source code, e.g. metrics on size of code based on line counting, being processed was a known concept in the art at the time the invention was made, e.g. LOC from Resource Standard Metrics as submitted in the application IDS. Further, Leonard, in a system using token analyzer analogous to the use of lexer/parser scheme by Scamp, discloses tokens-based measurements and lines of code derived from the token analysis and count for the lines of code derived from such measurement (e.g. col. 4, line 9 to col. 5, line 34). In case Scamp's software measuring system does not already provide a line of code counting tool, it would have been obvious for one of ordinary skill in the art at the time the invention was made to implement

such parser-based system by Scamp so that line of code technique as suggested by Leonard can be added because line of codes can be a factor determining how to predict resources, maintain, manage, or improve the state of a software product according to Leonard's approach of having a database for a life-cycle process associated with configuration management activity in view of standard determined by the RSM as mentioned above, particularly when it is admitted that code counting often encountered in large or complex software development and/or management systems to establish cost of the development and its implication in allotting resources or estimating the feasibility of the targeted product was a well-known concept based on said government or industry standard.

Nor does Scamp explicitly disclose that each set of configuration data to create token stream comprises keywords for one or more of the plurality of computer languages. But in view of the grammar file teachings by Scamp (*TOKEN COLLECTOR, ...specified in the grammar* - pg. 124-126 - Note: a grammar file necessarily entails the understanding of keywords in order to resolve a string of tokens), the grammar file being used per input source code and comprising keywords for the input language to enable token resolution is disclosed.

As for the limitation that one set of configuration data is used among a plurality of configuration data for one or more of a plurality of programming languages, the use of a dictionary for tokenizing a source language of a given programming language was a known concept in the compiler art at the time the invention was made. Duan, in method to provide grammar programming language to any input source code, and using lexical analysis and parser analogous to Scamp and token-based for code generation analogous to Leonard, discloses the use of a dictionary for a given source input file in order to create the lexical baseline and grammar

for that particular input source language; hence suggests more than one dictionaries being used for more than one of programming languages (e.g. Fig. 2A and related text; col. 4, lines 11-26). It would have been obvious for one of ordinary skill in the art at the time the invention was made to implement the token collection system by Scamp (combined with Leonard) so that for each input source of a given language, a configuration data set like the dictionary as suggested by Duan can be applied because this way it would alleviate the burden of overloading one single configuration data so to provision for more than one programming language; and would expedite the process of token creation and code lines metrics gathering.

As per claim 2, official notice is taken that at the time the invention was made source code being in programming language being in C++ or Pascal, Fortran, Java was a well-known concept for code analysis and measurement like that disclosed by Scamp. Therefore, it would have been obvious for one of ordinary skill in the art at the time the invention was made to implement to make known programming languages like C++ or Cobol as one of languages being subject for metrics collecting as suggested by Scamp because the purpose to collect metrics to a variety of languages would make the code measuring product as intended by Scamp more attractive.

As per claim 3, Scamp does not explicitly disclose a different set of configuration data from programming languages like C, C++, Java, Cobol, Basic or Fortran; but in view of the teachings by Duan to make the grammar parsing tool a multi-language applicable product and the intention by Scamp to implement the measuring tool onto a variety of languages as in claim 2, the providing of set of configuration data, e.g. dictionary, for more than one type of

programming language, e.g. C, C++, Fortran, Java, Cobol or Basic would also have been obvious using the corresponding rationale as set forth both in claim 1 and 2.

As per claim 4, the use of token based on some dictionary or grammar configuration data as disclosed by the combination Scamp/Leonard/Duan from above inherently differentiate tokens in different types represented by a value to the parser in order to enable correct syntax checking (e.g. parse tree) according to the language particular grammar rules; and this concept was a known concept in the art of compiler at the time the invention was made.

As per claim 5, Scamp discloses token type being a comment, an keyword or an identifier, or an operator (e.g. comment, identifier, keyword – from pg.; 124, 6th para to pg. 125, middle; Operator – pg. 127).

As per claim 6, a dictionary or a grammar configuration file assigning a type value inherently teaches maintaining such value consistent with the grammar specific to a particular language; and in view of the teachings by Scamp combined with Duan in claim 1 to impart a specific configuration data file(e.g. one dictionary per language) comprising specification for token type and token type value (re claim 4), the limitation of token type being invariable within one language grammar specification is also disclosed and obvious for the same rationale as to be able to make the measuring tool applicable for more than one language.

As per claim 7, the skipping of token not recognizable by a set of grammar rules during the parsing of token streams as taught by Scamp (e.g. skip over comment token or code structures written by natural language instead of programming languages – re claim 5) discloses portions of the token streams being ignored in generating the statement list because such portions are not in the language among the plurality of computer languages.

As per claim 13, Scamp discloses a method of code counting, comprising:

selecting one set of configuration data (e.g. *grammar file* - pg. 123, bottom – Note: a grammar is specific to one of the languages available, hence submitting one set reads on selecting one among possible other sets for other languages),

wherein collectively such plurality of sets are associated with a plurality of computer languages (e.g. *independently .. code is written* – pg. 127, 2nd para),

wherein such selected set comprises the keyword of a first computer language (e.g. *TOKEN COLLECTOR, ...specified in the grammar* - pg. 124-126 - Note: a grammar file necessarily entails the understanding of keywords in order to resolve a string of tokens),

parsing a first file in the first language into a first token stream in response to the selected set of configuration data (e.g. *TOKEN COLLECTOR, ...specified in the grammar* - pg. 124-126);

create a list of statements in response to the token stream (e.g. *section ... metrics ... local scope ... for each unique token* – pg. 124, 4th para; *token ... flag* - bottom pg. 124, top 8 paras - pg. 125 – Note: flagging code constructs as parsed using a grammar specification is equivalent to setting statements with tag, hence list of statements), at least some of the statements comprising a combination of two or more tokens (e.g. the report section having a cumulative type of metrics on tokens reads on comprising a combination of two or more tokens).

But Scamp does not explicitly disclose lines of code counting, and a count value in response to the list of statements. But Scamp disclose flagging based on token collector as well as terminal/non-terminal tokens (re claim 1). Thus, these limitations would have been obvious by virtue of the corresponding rejection as set forth in claim 1 from above.

Nor does Scamp explicitly disclose selecting among a plurality of sets of configuration data, each associated with a programming language; but this limitation has been addressed in claim 1 above using Duan's teachings.

As per claims 14-18, these claims correspond to claims 2, 4-6, and 3 respectively, and are rejected using the corresponding rejection as set forth therein.

As per claim 23, Scamp discloses a computer readable medium storing a software counting tool, comprising:

a configuration file for one or more of a plurality of computer languages (e.g. *grammar file* - pg. 123, bottom - Note: a grammar file is specific to possibly one of the languages available, hence submitting one file reads on selecting one among possible other files for other computer languages);

a tokenizer to parse a first language source code into a token stream(*TOKEN COLLECTOR, ...specified in the grammar* - pg. 124-126);

wherein a tokenizer parses source code written in any of the plurality of computer languages (e.g. Figure on pg. 123; *independently .. code is written* – pg. 127, 2nd para; *TOKEN COLLECTOR, ...specified in the grammar* - pg. 124-126);

wherein a tokenizer creates a token stream in response to the configuration file associated with the first computer language(e.g. *TOKEN COLLECTOR, ...specified in the grammar* - pg. 124-126);

a statement builder creating a list of statements in response to the token (e.g. *section ... metrics ... local scope ... for each unique token* – pg. 124, 4th para; *token ... flag* - bottom pg.

124, top 8 paras - pg. 125 – Note: flagging code constructs as parsed using a grammar specification is equivalent to setting statements with tag, hence list of statements),

at least some of the statements comprising a combination of two or more tokens (e.g. the report section having a cumulative type of metrics on tokens reads on comprising a combination of two or more tokens).

But Scamp does not disclose source code line counting system nor does Scamp disclose a counter to generate a value in response to the list of statements. But these limitations have been addressed in claim 1 from above.

Nor does Scamp explicitly disclose a plurality of configuration files, each associated with one or more of a plurality of computer languages; but this limitation has been addressed in claim 1 above using Duan's teachings.

As per claim 24, Scamp in combination of Leonard and Duan, discloses a plurality of additional statement builders each associated with one or more computer languages and operable to generate a statement list in response to a token stream generated from a source file associated with one or more languages (Note: if Scamp is providing a tool to cooperate with a plurality of languages, the creation of more set of code statements per grammar file (in view of Leonard's teachings) in conjunction with a token stream derived therefrom would be implicitly disclosed as per the rationale of claim 1).

As per claims 25-26, refer to claims 4 and 6, respectively.

As per claim 37, Scamp discloses a code counting system, comprising:

a set of configuration data associated with at least one computer language (e.g. *grammar file* - pg. 123, bottom), a plurality of which set collectively associated with different computer languages (e.g. *independently .. code is written* – pg. 127, 2nd para);

a computer readable medium storing a software counting tool operable to

receive a source code file (Figure pg. 123); and

parsing the source file to create a token stream in response to the selected set of configuration data (e.g. *TOKEN COLLECTOR, ...specified in the grammar* - pg. 124-126; *source file* - Figure of pg. 123);

create statements in response to the token stream (e.g. *token ... flag* - bottom pg. 124, top 8 paras - pg. 125 – Note: flagging code constructs as parsed using a grammar specification is equivalent to setting statements with tag, hence list of statements),

at least some of the statements comprising a combination of two or more tokens (Note: the generation of flags with line of code structures based on tokens reads on statement list comprising a combination of two or more tokens); in response thereto compute a statistical measure (e.g. *into data structures* – pg. 123 bottom; *token flag* - pg. 124-125; pg. 125; Figure of pg. 123).

But Scamp does not disclose source code line counting system nor does Scamp disclose a statistical measure related to the number of source code lines in response using the one of the sets configuration data. But these limitations have been addressed in claim 1 from above using Leonard.

But Scamp does not explicitly disclose a plurality of set of configuration data. But this limitation has been addressed in claim 1 using Duan.

4. Claims 8-11, 19-21, and 27-28 are rejected under 35 U.S.C. 103(a) as being unpatentable over IBM, "Software Compiler for Analysis and Measuring Programs", 9/1993 (i.e. Scamp), in view of Leonard, USPN: 5,729,746 and Duan et al., USPN: 6,529,865, as applied to claims 1, 13, 17, 25, and further in view of Greenfeld, USPN: 4,931,928 (hereinafter Greenfeld).

As per claims 8-11, validating code constructs by identifying them by correct sequences being identifiable via a parsing process was a well-known concept at the time of the invention, i.e. the use of a parser to segregate tokens in valid and identifiable sequences for a given grammar was an inherent process in compilers when the token streams is analyzed using a parse tree so as to validate the constructs of the source code with such constructs being assigned with specific identification. Thus, the teachings of a parser by Scamp/Leonard or Duan implicitly discloses (re claim 8) statement being categorized in different types and that (re claim 11) a relationship between tokens in a parse tree is examined with respect to other tokens in the stream when the code statement is created and verified for correctness. The assigning of a type value for each code statement type would also have been obvious in light of the well-known concepts and the rationale as set forth in claim 5 from above.

But Scamp does not explicitly teach that the statement type value does not vary based on the computer language of the source code of the first file (re claim 9), that the type is either data, executable or compiler (re claim 10). The rationale as to maintain a statement type to be invariable within a grammar rule sets of a specific programming language would have been obvious herein using the same rationale as set forth in claim 6 above. Further, Greenfeld, in a method to provide reuseable set of language configuration data for program code analysis analogous to the grammar file by Scamp or to the dictionary by Duan, discloses Semantics

information enabling extraction of code constructs and differentiation the type being extracted as code symbols, control flow information or data and relationships information between type (e.g. col. 5, line 30-44). It would have been obvious for one of ordinary skill in the art at the time the invention was made to implement the differentiation of code statement in type as suggested by Scamp/Leonard/Duan's parsing process such that the type can be code, data, and compiler control directives as suggested by Greenfeld because this way code statements can be distinguished easily so to make efficient runtime resources allotment and also can be persisted based on its category or type enabling a database of information to be reused for further analysis as mentioned by Greenfeld.

As per claims 19-20, these claims correspond to claims 8 and 9, respectively and are rejected using the corresponding rejection as set forth therein.

As per claim 21, this claim includes the limitations of claim 8 and 9, from above, and is rejected using the corresponding rejections as set forth therein.

As per claims 27-28, these claims correspond to claims 8 and 9, respectively and are rejected using the corresponding rejection as set forth therein.

5. Claims 12 and 22 are rejected under 35 U.S.C. 103(a) as being unpatentable over IBM, "Software Compiler for Analysis and Measuring Programs", 9/1993 (i.e. Scamp), in view of Leonard, USPN: 5,729,746 and Duan et al., USPN: 6,529,865, as applied to claims 1, 13; and further in view of Greenfeld, USPN: 4,931,928 (hereinafter Greenfeld), and Bloom, USPN: 3,711,863 (hereinafter Bloom).

As per claim 12, Even though Scamp teaches generating list of statements for 1st or 2nd instance o of token stream (re claim 1 – *Report ... section*, pg. 124, 4th para), Scamp does not

disclose creating a second list of statements from the token stream generated from parsing a second source file of a different version than the first source file; and comparing the first list of statements to the second list of statements. However, code being developed within a life-cycle encompassing code change management is suggested by Leonard (Fig. 1-3); and the merging of code based on a baseline code to extract differences among other developer's code instances is suggested by Greenfeld (e.g. col. 7, lines 35-53). Further, Bloom, in a method teaching code comparator technique such as that suggested by Greenfeld or any code merging in software configuration management, discloses comparing two files to extract the difference and save the number of delta lines (e.g. step 54 - Fig. 2C). It would have been obvious for one of ordinary skill in the art at the time the invention was made to manage the code analysis and software development by Scamp/Leonard in view of Greenfeld's code difference extraction, so as to implement a second list of statements as from a token stream derived from a second file, a delta extraction of code differences as taught by Bloom, and record a count responsive to the differences from comparing the first list of statements and the second list of statements according to the scheme of token generation as addressed in claim 1. The motivation is that this would enable to store a difference between code versions in terms of a value which can be persisted according to Greenfeld's baseline persisting method and life-cycle code analysis by Scamp/Leonard, thus providing numerical representation of the difference as taught by Bloom between two versions of code being managed under the management tool by Scamp/Leonard (enhanced by the baseline code comparing by Greenfeld).

As per claim 22, refer to rejection of claim 12.

6. Claims 29-36 are rejected under 35 U.S.C. 103(a) as being unpatentable over IBM, "Software Compiler for Analysis and Measuring Programs", 9/1993 (i.e. Scamp), and Leonard, USPN: 5,729,746; and further in view of Greenfeld, USPN: 4,931,928 and Bloom, USPN: 3,711,863 (hereinafter Bloom).

As per claim 29, Scamp discloses a measuring method comprising

parsing a first file to create a token stream (refer to rejection of claim 1);

creating a first list of statements in response to the token stream;

creating a second list in response to a second token stream from parsing a second file (e.g. *section ... metrics ... local scope ... for each unique token* – pg. 124, 4th para – Note: Report generated by token stream in conjunction with each instance of grammar file and token collector reads on first list of statement and second list of statements; *token ... flag* - bottom pg. 124, top 8 paras - pg. 125 – Note: flagging code constructs as parsed using a grammar specification is equivalent to setting statements with tag, hence list of statements);

at least some of the statements in the first list comprising a combination of 2 or more tokens; at least some of the statements in the second list comprising a combination of 2 or more tokens (Note: the creation of report with sections having a cumulative token account representations per instance of stream reads on first list having combination of tokens and 2nd list having combination of tokens, a metrics statement being/representing a combination of tokens)

But Scamp does not disclose comparing the first list to the second list of statements to generate one count responsive to such comparison. The teachings to provide software analysis with LOC count generating in a life-cycle code management is taught by Leonard (re claim 1) and the providing of code baseline for merging or identifying code version differences has been

taught by Greenfeld (re claim 12). Further, the limitation as to compare files to generate a count based on the differences between the files would have been obvious in view of Bloom's counting of delta lines as set forth in the rationale in claim 12 above combined with the teachings of the combined Scamp/Leonard/Greenfeld.

As per claims 30-33, merging tools used to exhibit the differences between 2 files set side-by-side implicitly disclose accounting for the lines of source code that are unchanged or changed for each file being submitted in the merging tool, and this concept was a well-known concept at the time the invention was made. Using the teachings by Bloom to count the differences, the limitations as to account for the number of unchanged/changed lines in one file with respect to the other file among the files being compared would have been but one of the variations from making use of the results out of the delta extraction as suggested in the counting by Bloom or many file merging tools. Hence, it would have been obvious for one of ordinary skill in the art at the time the invention was made to implement the merging tool as suggested by Greenfeld or the comparator by Bloom so that (re claim 30)the number of statements modified in the 2nd file with respect to the first file, (re claims 31-32) the number of statements in one file but not in the other, and (re claim 33) the number of statements commonly present in both files would also be generated or computed depending on the needs for gathering information representing how the changes to the files are to be persisted and recorded for future analysis, according to the software configuration management or versioning so well-known as suggested by Leonard or Greenfeld.

As per claim 34, only Greenfeld suggests comparing against a modified version of the first file (e.g. col. 7, lines 35-53); and the limitations in this claim are the same as recited in the

claims 30-33, respectively. In light of the rationale used in the rejection from claims 30-33, this claim is also rejected based on the corresponding rejection as set forth therein.

As per claim 35, this claim is a computer-readable medium claim of claim 29 above, hence is rejected using the rejection as set forth therein.

As per claim 36, this claim corresponds to claim 34, hence is rejected using the same rationale as set forth therein.

Response to Arguments

7. Applicant's arguments filed 9/30/2004 have been fully considered but they are either not persuasive, or moot in light of the new rationale and/or new grounds of rejection. Following are the Examiner's observations thereto.

(A) Applicant have submitted that the Action relies of Leonard for disclosing 'creating a list of statements ... stream' (Appl. Rmrks, pg. 12, 2nd from the bottom para); and that Scamp does not teach or suggest source code count, but merely discloses token counts by producing a report, and is silent about generating a list of statements (Appl. Rmrks, pg. 13, top para). First, Leonard is used to support the fact that code count is line of code counting among other metrics; and Applicant fail to show what is in the claim that specifically distinguish over the references used so to render the rationale to combine Leonard's teachings with the code count by SCAMP inappropriate or non-obvious.

Second, the rejection has provided two ways to look at the limitation recited to as 'list of statements'; i.e. 2 alternatives to read on such limitation. One alternative is provided via the statements in the report including association of tokens statistics with how tokens are distributed in the code being parsed. The other alternative is to enclose the flag setting in parallel with code

constructs, and as a whole, such flag association per code elements being parsed based on some predetermined significance amounting to a strong suggestion to, if not a teaching for, said listing of statements. In case Scamp does not organize the flags according to a listing process as dictated by the grammar file, there is sufficient motivation to do so; and the rejection has provided the rationale as to make such listing obvious when Scamp provide parsing of line of code based on token and user specification so that the flags enable the count as desired. The Applicant's observation as to why Scamp is silent on such list of statements is not directed at the state of the current rejection; hence is moot.

(B) Applicants have submitted that Leonard does not provide metrics relating to developed source code; but only in anticipation of what the code will exhibit (Appl. Rmrks, pg. 13, 2nd para). A USC 35 103 (a) rejection only needs to show a particular problem the inventor aims at in his endeavor; and so long as the prior art is also addressing the same problem so as to shed sufficient grounds for an ordinary skill in the art to see obviousness in reaching a result trying to solve that problem as commonly endeavored by the inventor and the prior art, the rejection is proper regardless whether the prior art used is strictly in the same field of the invention or not. The endeavor is code counting and the problem is that a count needs to be achieved, e.g. count of lines, modules, chunk of lines or code constructs. Suppose Scamp also addresses code counting but via token measurements, the problem to solve is a count to be achieved; and both Scamp and Leonard are reaching a count, one by way of tokens being flagged, the other by way of lines of code. The fact that allegedly Leonard's code measurement is not falling in the field of programming code being developed is not the particular problem as mentioned above; hence the argument is misplaced to overcome the rejection.

(C) Applicants have submitted that Leonard does not disclose line of code count but merely approach it with token counts (Appl. Rmrks, pg. 13, 3rd para). Again, the endeavor is code count, and the problem is to achieve a count. Even though, as alleged, Leonard approaches this endeavor by means of token counting, which very commonly known when a parsing takes place in processing a source file, the problem is providing a count aiming at the original endeavor of LOC counting. Again, so long as code counting is the major endeavor, achieving a count by means of tokens via tokens only or via line-by-line measurement are only achieving a problem with a result. More importantly, the claim as recited does not preclude that because it is lines of code counting, any form of approach to achieve this count would not be or cannot be through counting of tokens. Leonard is used to provide the fact that code counting endeavor can achieve a result via counting in a LOC context; and if line counting is not expressly taught by Scamp, the suggested LOC approach by Leonard would provide sufficient grounds to combine the references to make the line of code counting obvious in the context of solving a particular problem as mentioned above.

(D) Applicants have submitted that the references by Scamp, Leonard does not teach or suggest 'create a list of statements ... of two or more tokens' and 'generate a count value ... list of statements' (Appl. Rmrks, pg. 15, bottom 2 paras). The rejection has provided rationale as to why Scamp has disclosed the list of statements and why that would be obvious. Refer to section A above.

(E) Applicants have submitted that the office action simply relies on hindsight (Appl. Rmrks, pg. 16, 2nd para) and that Leonard and Scamp teach different endeavors (pg. 16, bottom). The rejection does not need to provide motivating evidence or suggestion from the references so

long as such suggestion has been recognized as a well-known concept. At the time the invention was made, one skill in the art would not have agreed that line of code count happened to be novelty above other methodologies proposed to provide code metrics. It is admitted that code counting often encountered in large or complex software development and/or management systems to establish cost of the development and its implication in allotting resources or estimating the feasibility of the targeted product was a well-known concept based on a government or industry standard. A rationale to provide obviousness needs to be contemplated in light of the whole prior art/invention as well as properties inherent to the endeavor/domain pertinent thereto; and code count has been known to belong to inherently taught software endeavor for cost estimates. In the light that Scamp is but a tool to provide such code counting in view of the additional LOC approach by Leonard, the whole endeavor to provide a count is fulfilled by the combination via which Leonard suggests line of code and Scamp provide token based counting tool. Along with the fact that Leonard just like any software development life cycle teaches a support database (Fig. 3), in combination with the code count endeavor in light of engineering industry well-recognized practices and code count standard, it is not required that the references provide explicit a suggestion leading to the motivation to combine when such motivation comes from a industrial standard or admitted practices. One skill in the art would not need to glean from the invention as asserted by the Applicants in order to discover that code count via line of code counting is novelty beyond any practice proffered so far by software development evaluation techniques or various code counting tools/products. That Scam and Leonard amount to different endeavors and thus rendering the combination unobvious, this argument has been addressed above in sections B and C above.

The claims stand rejected as set forth in the rejection.

Conclusion

8. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (272) 272-3735. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kakali Chaki can be reached on (571)272-3719.

The fax phone number for the organization where this application or proceeding is assigned is (571) 273-3735 (for non-official correspondence – please consult Examiner before using) or 703-872-9306 (for official correspondence) or redirected to customer service at 571-272-3609.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

VAT
January 28, 2005

KAKALI CHAKI
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100